SYSTEM AND METHOD FOR MEMORY INTERLEAVING USING CELL MAP WITH ENTRY GROUPING FOR HIGHER-WAY INTERLEAVING

The Field of the Invention

5   This invention relates generally to cell maps for computer systems. This invention relates more particularly to a system and method for memory interleaving using cell maps with entry grouping for higher-way interleaving.

Background of the Invention

10   Historically, main memory was physically situated on a central bus. Within this type of system, memory requests consisting of full physical addresses, were forwarded to the memory subsystem and the data was returned. In a distributed memory system, main memory is physically distributed across many different cells. A cell may consist of a number of processors, an

15 input/output (I/O) device, a cell controller, and memory.

        In a distributed system, memory can be non-interleaved or interleaved. Prior art systems of and methods for interleaving memory are described and set forth in, for example, U.S. Patent No. 5,530,837, entitled METHODS AND

20 APPARATUS FOR INTERLEAVING MEMORY TRANSACTIONS INTO AN ARBITRARY NUMBER OF BANKS, issued June 25, 1996 to Williams *et al.* and U.S. Patent No. 5,293,607, entitled FLEXIBLE N-WAY MEMORY INTERLEAVING, issued March 8, 1994 to Brockmann *et al.* U.S. Patent Nos. 5,530,837 and 5,293,607 are assigned to the owner of the present invention, and

25 both patents are incorporated herein by reference in their entireties. In a non-interleaved access method wherein memory is divided into or across multiple physical cells, a unified, contiguous block of memory space is addressed by first sequentially accessing all memory of a first cell followed by sequential access of all memory available in a second cell, etc. If each cell has been configured with

30 its maximum amount of possible memory, the memory will appear, and be addressed as one contiguous memory block to the system. However, if not every

cell is configured to its maximum memory capability, this non-interleaved scheme may result in holes within the memory space corresponding to missing memory blocks within the cells. Non-interleaved memory also requires multiple, sequential accesses to a particular cell since both instructions and data

5   tend to be used sequentially. While a benefit when stored locally, a processor continuously or frequently accessing a remote memory in another cell consumes significant overhead including processing and communications resources at both the local and remote cells and the connecting network(s). When substantially continuous, these resources may become unavailable to other processes and

10   degrade system performance.

Alternatively, the memory within a distributed memory system can be accessed through an interleaving protocol. Interleaving memory across several cells allows for more uniform access to memory. For example, if a system includes two cells that are connected together through a bus system, with each

15   cell including memory and four separate processors, by interleaving the memory in cell 1 with the memory in cell 2, all eight processors in the system have a more uniform access to each memory location. Interleaving memory across the two cells also ensures consistency in latency delays for each processor in accessing memory locations. Interleaving memory across the two memory

20   locations also reduces the possibility of bottlenecks when processors attempt to access or retrieve information from memory.

As an example of interleaving, assume that the memory contained within a system is distributed across four cells labeled 0, 1, 2, and 3. Further assume that each of cell 0 and cell 1 contain eight gigabytes (GB) of memory, while

25   cells 2 and 3 each contain four GB of memory. The overall system therefore contains twenty-four GB of memory. The distributed memory could be interleaved as follows. Since each of the four cells contains at least four GB of memory, the first interleave entry, entry 0, would contain four GB of memory from each of cell 0, 1, 2, and 3 for a total of sixteen GB of memory. All of the

30   memory available in cell 2 and cell 3 have now been used in interleave entry 0.

Cell 0 and cell 1 each contain four GB of unused memory. Interleave entry 1 would contain the four GB of memory from cell 0 and the four GB of memory from cell 1. Interleave entry 1 therefore contains eight GB of memory, four from cell 0 and four from cell 1. The twenty-four GB of memory in the four

5   cells have now been broken out into two interleave entries. The twenty-four GB of memory from the four cells can now be viewed as one contiguous block as follows. GB 0 through 15 are located in the lower half of cells 0, 1, 2, and 3, GB16 through 23 are located in the upper portion of cells 0 and 1. This interleaving occurs at the cache line level. To a processor, the twenty-four GB

10   of information appears to be one contiguous block. While the twenty-four GB of information appears to be one contiguous block, physically the twenty-four GB of information is distributed among four different cells.

Large computer systems have used cell maps to distribute processor accesses around to avoid hot spots. Cell maps have been used to find destination

15   modules for the cell/node based systems. Cell maps have been used to interleave memory across cells to provide a more uniform access pattern to the memory. Cell map entries have been used to provide one, two, four, eight, sixteen, thirty-two, and sixty-four way interleaving. The size of cell map entries determines how many ways the memory is interleaved.

20   In previous systems that used cell maps, the number and size of cell map entries had to be predetermined. To do higher-way interleaving, the cell map structures had to be designed up-front accordingly. The number of ways of interleaving that can be used is dependent on the size of the system, memory configuration and other parameters. More entries with more ways of interleave

25   use more resources. In some previous systems, there had been a need for higher-way interleaving, but the system resources were limited, requiring lower-ways of interleave per entry, or reserving only a subset of the entries for higher-way interleaving, with the remainder of the entries providing lower-way interleaving. In previous systems, the cell maps were not always used very efficiently, and

30   were not scaleable to provide more ways of interleaving.

It would be desirable to achieve higher-ways of interleaving by combining lower-way interleave entries.

Summary of the Invention

5   One form of the present invention provides a method of accessing a plurality of memories in an interleaved manner using a contiguous logical address space. At least one map table is provided. The at least one map table includes a plurality of entries. Each entry includes a plurality of entry items. Each entry item identifies one of the memories. A first logical address is

10   received. The first logical address includes a plurality of address bits. The plurality of address bits includes a first set of address bits corresponding to a first set of entries in the at least one map table. A first entry in the first set of entries is identified based on the first set and a second set of the address bits. A first entry item in the first entry is identified based on a third set of the address bits.

15   The memory identified by the first entry item is accessed.

Brief Description of the Drawings

Figure 1 is a block diagram illustrating a node including four cells and a crossbar.

Figure 2 is a block diagram illustrating a distributed memory system,

20   with the memory distributed across four cells.

Figure 3 is a table illustrating four-way and eight-way interleaving.

Figure 4 is a table illustrating three-way interleaving.

Figure 5 is a block diagram illustrating a sixty-four-way interleave cell map entry for single cache line interleaving.

25   Figure 6 is a chart identifying distinguishing characteristics among interleaving groups.

Figure 7 is a block diagram of interleaving across cells containing non-power-of-two memory amounts.

4

Figure 8 is a block diagram of interleaving across three cells.

Figure 9 is a diagram of a cell map for providing memory interleaving according to one embodiment of the present invention.

Description of the Preferred Embodiments

5   In the following detailed description of the preferred embodiments, reference is made to the accompanying drawings, which form a part hereof, and in which is shown by way of illustration specific embodiments in which the invention may be practiced. It is to be understood that other embodiments may

10   be utilized and structural or logical changes may be made without departing from the scope of the present invention. The following detailed description, therefore, is not to be taken in a limiting sense, and the scope of the present invention is defined by the appended claims.

Figure 1 is a block diagram illustrating a processing system or node 100,

15   which includes four cells 105, 110, 115 and 120, connected via crossbar 125. Each cell has a corresponding block of memory -- memory 130 in cell 105; memory 135 in cell 110; memory 140 in cell 115, and memory 145 in cell 120. Each cell also includes four processors (numbered 150, 155, 160 and 165 for cell 105), an input/output (I/O) module (numbered 170 for cell 105), and a cell

20   controller (numbered 175 for cell 105).

Each of the four processors 150 through 165, and the I/O module 170 require access to memory 130. Cell controller 175 is the interface between both the I/O module 170 and the processors 150 through 165, and the memory 130. In order for a memory-accessing device (e.g., any of the processors 150 through

25   165, or the I/O module 170) to access a correct portion of memory, the cell controller 175 translates a logical memory address known to the memory-accessing device into a physical address. The physical address allows cell controller 175 to know how to route the memory request. Any of the memory-accessing devices in cell 1 could also access cell 115's memory 140, cell 120's

30   memory 145, or cell 110's memory 135. Cell controller 175 uses a cell map to

translate a logical memory address from a memory-accessing device into the proper physical address that can be used to access an appropriate memory. In one embodiment, cell controller 175 contains a different cell map for each memory-accessing device. For the embodiment illustrated in Figure 1, cell

5   controller 175 would contain five different cell maps, one for each attached memory-accessing device.

One embodiment of a cell map is a table with a plurality of rows and a plurality of columns, where each row corresponds to an interleaving entry, and each column within a row corresponds to an entry item that identifies one of the

10   cells in the system.

Figure 2 shows a distributed memory system 200 with memory distributed over four cells: cell 0, cell 1, cell 2 and cell 3. Cell 0 contains a total of eight gigabytes (GB) of memory; cell 1 contains six GB of memory; cell 2 contains four GB of memory and cell 3 contains two GB of memory. In one

15   embodiment, interleaving over these four cells is accomplished in the following manner. First, an assessment is made as to which cell has the minimum amount of memory available for interleaving. In this case, cell 3 only contains two GB of memory. Interleaving entry 0 would therefore interleave two GB of memory from cells 0, 1, 2 and 3 for a total of eight GB of memory. The resultant

20   interleaving entry is shown in row 305 of Figure 3, wherein each row identifies up to eight cell memories. Above row 305 are eight three-bit combinations, with each three-bit combination being associated with one of the eight entry items in row 305 (and row 310). These three-bit combinations are used for identifying a particular entry item, and are discussed in further detail below.

25   Next, an assessment is made to determine the minimum amount of memory left in any available cell. In this case, it would be two GB in cell 2. Interleaving across cell 0, cell 1 and cell 2 is provided by using three interleaving entries as shown in Figure 4. Interleaving entries 1, 2, and 3 (the first three rows, respectively, under the three-bit combinations in Figure 4) are used for the

30   second two GB block of cell 0, the second two GB block of cell 1, and the top

two GB portion of cell 2. At this point, interleaving entry 0 addresses a total of eight GB of information, while interleaving entries 1, 2 and 3 each contain two GB of information. Cell 3's memory resources were completely used in interleaving entry 0. Cell 2's memory resources were exhausted through the

5   completion of interleaving entry 3.

The next step in this process is to identify the remaining memory left to be allocated in any of the cells. In this case, two GB of memory are left in cell 1. Interleaving entry 4 would normally contain two GB of memory from cell 0 and two GB of memory from cell 1. This arrangement would add four GB to the

10   already mapped fourteen GB, for a total of eighteen GB. However, it is preferable in one embodiment to start each table entry on an integer multiple of the group size, *i.e.*, at sixteen GB. Thus, two GB needs to be added next to allow the four GB in cells 0 and 1 to occupy one interleaving entry. Therefore, interleaving entry 4 (not shown) represents and maps to the uppermost two GB

15   that reside in cell 0. Now the interleaving entries defined thus far (0, 1, 2, 3 and 4) contain sixteen GB of memory. Interleaving entry 5 (not shown) maps to the remaining two GB that reside in cell 1 and the remaining two GB in cell 0. This is permissible because the sixteen GB allocated so far in entries 0, 1, 2, 3 and 4 is a multiple of four GB. In summary, the twenty GB of memory that is

20   contained in cell 0 through cell 3 is now contained in 6 interleaving entries. The first interleaving entry contains eight GB of information, two GB from each of the four cells. Interleaving entries 1, 2, 3 and 4 have two GB contained within them, and interleaving entry 5 has four GB.

An interleaving group is defined as a unit of a plurality of cells that have

25   the same amount of memory confined for interleaving in a given memory range. An interleaving group is composed of one or more interleaving entries. Interleaving group 0 is composed of the eight GB that reside within interleaving entry 0, or in other words, two GB from each of the four cells. Interleaving group 1 is composed of the six GB that reside within interleaving entries 1, 2 and

30   3, or in other words, two GB from cell 0, two from cell 1 and two from cell 2.

Interleaving group 2 is composed of the uppermost two GB from cell 0 (i.e., interleaving entry 4). Interleaving group 3 is composed of two GB from cell 0 and cell 1 (i.e., interleaving entry 5).

To identify the specific cell that the desired physical memory is located

5   in, a first set of address bits from an incoming memory address are used to identify a row in the cell map, and a second set of address bits from the incoming memory address are used to index into the identified row and identify an entry item. In one embodiment, forty-four bit memory addresses are used, with address bits twenty-nine through forty three corresponding to the first set of

10   address bits, and bits six through eight corresponding to the second set of address bits. Three bits (e.g., bits six through eight) are used for indexing into entries with eight columns or entry items. For entries with sixteen entry items, four bits (e.g., bits six through nine) are used for indexing. Similarly, for each doubling of the entry size, an additional indexing bit is used.

15   Interleaving is most effective when the amount of consecutive memory accessed is kept small. In interleaving cache lines, the cell map indicates which cells the cache lines are located on. So referring back to interleaving entry 0 (row 305 in Figure 3), if the cache lines were interleaving between cells 0, 1, 2 and 3, row 305 indicates how the cells would be represented within the table. In

20   one embodiment, the three bits in the plurality of three-bit combinations above row 305 correspond to address bits six through eight of an incoming logical memory address. These three bits of an incoming address are used to index into the row and identify one of the entry items. Each entry item in row 305 identifies one of the four cells. As shown in Figure 3, if bits six through eight of

25   a memory address are 000, that indicates to the cell controller that the physical address for the memory is in cell 0. A value of 001 for bits six through eight of a memory address indicates to the cell controller that the physical address for the memory is in cell 1. Similarly, each of the other six possible values for bits six through eight of a memory address is used by the cell controller to identify a cell

30   from the table.

Referring again to Figure 1, a single node with four cells is shown. Each cell has its own memory device. If, however, the overall system had two four-celled nodes, there would be eight cells available, each cell with its own memory. If these cells were labeled cell 0 through cell 7, the memory could be interleaved across all eight cells. Row 310 of Figure 3 shows how memory could be interleaved over all eight cells. In this case, the three-bit designator (bits six through eight of the memory address) indicates which of the eight cells contains the physical address of the memory. Thus, row 310 illustrates eight-way interleaving, and row 305 illustrates four-way interleaving.

If a system has a three-cell configuration, the cell map would appear as illustrated in Figure 4, which illustrates three-way interleaving. The three cells would effectively be mapped over three rows in the cell map. The first column of the first row would contain a value for cell 0. The second column of the first row would contain a value for cell 1. And the third column of the first row would contain a value for cell 2. This sequence would be repeated until the last cell ended up in the last column of the third row.

A row in a cell map is identified by the cell controller through a combination of a mask, a comparator, and memory address bits from the memory-accessing device. With the mask, bits of the address that are irrelevant to the determination of the appropriate row are masked out. The comparator compares the masked address portion with match values to identify a corresponding row of the cell map.

Figure 5 is a block diagram illustrating a sixty-four-way interleaved cell map entry for single cache line interleaving. Since the cell map entry in Figure 5 is sixty-four-way interleaved, six bits (e.g., bits six through eleven) of a memory address are used for indexing into the entry. Entry enable block 510 identifies an appropriate row of the cell map using address portion 505, which includes bits twenty-nine through forty-three of an incoming memory address. An address range is determined from address portion 505, which is used to select a row. As shown in Figure 5, entry enable block 510 performs the following Equation I:

9

Equation I

Entry_enable = ((ADDR AND MASK) == cmp)

As shown by Equation I, entry enable block 510 masks address portion 505 to discard the least significant bits of the address, and compares the masked 5 address to "cmp," which is a comparison value or match value for the row. If the values in the comparison are equal, Entry_enable is assigned a logical true value, indicating a matching row. If the values are not equal, Entry_enable is assigned a logical false value, indicating a non-matching row.

Once a row is selected, bits six through eleven of the memory address are 10 used to identify one of the sixty-four entry items within that row of the cell map, and a "CELL" and "Cell_Addr" are determined and output. "CELL" represents a cell ID for uniquely identifying cells, and "Cell_Addr" represents a cell address, which is described in further detail below. As shown in Figure 5, address portion 515 includes address bits six through eleven and twenty-nine 15 through forty-two, which are used to determine the appropriate cell ID and cell address. Additional table entry items can be selected through additional inputs as shown by address input 520. As shown in Figure 5, a column is selected by performing a comparison like that shown in the following Equation II:

Equation II

20 Column_Select = (ADDR[11:6] == 0x0)

As can be seen from Equation II, bits six through eleven of the memory address are compared to a hexadecimal value corresponding to the column (e.g., 0x0 in this case, which corresponds to the first column). If the values being compared are equal, Column_Select contains a logical true value indicating a 25 match. If the values being compared are not equal, Column_Select contains a logical false value indicating no match.

The use of the comparator and the mask in entry enable block 510 will be described in further detail with reference to Figure 6. Figure 6 shows a binary count representing decimal numbers from 0 to 16. The decimal range from 0 to

10

16 has been broken up into four different groups: Group 605 containing

numbers 0 through 3; group 610 containing numbers 4 through 7; group 615

containing numbers 8 through 11; and group 620 containing numbers 12 through

15. In order for the mask to be used efficiently, a way to identify the different

5   groups is necessary. In this case, it can be seen that the first two bits, or the two

most significant bits (the two left-most bits) can be used to differentiate between

the various groups. The four numbers displayed in group 605 each start with 00;

each of the numbers in group 610 start with 01; each of the numbers in group

615 start with 10; and each of the numbers in group 620 start with 11. A mask is

10   therefore established that consists of 1100. For each bit of interest a "1" is

placed, and for each bit not of interest, or "don't care" a "0" is placed. The

comparator is set equal to the starting address of the memory block. The mask is

used to decide in which of the four groups, 605 through 620, the desired

sequence of bits resides. A comparison between the two most significant bits

15   will determine a specific row within the cell map.

To further illustrate the use of the comparator and the mask, several

examples will be examined. Referring to Figure 7, a node contains four cells,

where cell 0 has five GB, cell 1 has three GB, cell 2 has three GB, and cell 3 has

three GB of memory. In hexadecimal notation, one GB equals

20   0X000_40000000. Interleaving entry 0 of Figure 7 would contain two GB from

each of the four cells for a total of eight GB of memory. This memory block

would go from zero GB up to eight GB. The comparator would be equal to

0X000_00000000. The value for the mask would be determined by identifying

the bits that need to be examined to decide if the memory location is greater than

25   eight GB. In each bit position that must be examined to decide if the value is

over eight GB, a "1" would be placed. So in hexadecimal notation, the mask

would be equal to 0Xffe_00000000. Since bits represented by the eight least

significant numbers in the hexadecimal notation are not necessary to determine

whether the value of the memory location is over eight GB, they do not need to

30   be examined. The mask therefore only contains the bit positions that need to be

examined to decide if the memory value from one of the memory accessing-

devices is over eight GB or not. With this mask and comparator established, any access in the range of zero to eight GB will activate only interleaving entry 0. Within the appropriate row of the corresponding cell map, the cell numbers may be identified as 0, 1, 2, 3, 0, 1, 2, 3. This is similar to row 305 in Figure 3. This

5 configuration would achieve a four-way interleaving between cells 0 through 3. For interleaving entry 1, again all four cells are used, but this time, only one GB of memory from each cell is used. In this case, the comparator value is equal to 0X002_00000000, which is equivalent to eight GB. This is the starting value for this interleaving entry. The mask value in this case is 0Xfff_00000000.

10 Once interleaving entry 1 has been established, two GB remain in cell 0.

In general, interleaving is performed over the largest blocks first and then subsequent interleave blocks are smaller in size. The two GB remaining in cell 0 would normally be broken up into one GB chunks as shown in Figure 7. However, in this case, the twelve GB contained in interleaving entry 0 and

15 interleaving entry 1 is divisible by the two GB remaining in cell 0. Therefore, interleaving entry 2 will contain the 2 GB of memory left in cell 0 and the comparator value will be 0X003_00000000, which is twelve GB, and the mask value will be 0Xfff_80000000.

As a final example, if, as shown in Figure 8, a node contained three cells,

20 each of which contained two GB of memory, interleaving would be performed as follows. For interleaving entry 0, the comparator would be 0X000_00000000, or zero GB. The mask would be 0Xfff_80000000. This allows interleaving between the first GB of cell 0 and the first GB of cell 1. The row entry for this case would consist of 0 followed by 1, alternating throughout the length of the

25 row. This interleaves two ways over the first two GB of memory, again, one GB from cell 0 and one GB from cell 1.

For interleaving entry 1, the comparator value would be 0X000_80000000, or equal to two GB. The mask value for this would be 0Xfff_80000000. The corresponding cell map row would be alternating 1's and

2's throughout the length of the row, which allows interleaving two ways over the second GB from cell 1 and the first GB from cell 2.

Interleaving entry 2 has a comparator value of 0X001_00000000, which is equal to four GB. The mask value is 0Xfff_80000000. The corresponding

5   row within the cell map for this interleaving entry would be alternating 2's and 0's throughout the length of the row. This interleaving entry interleaves two ways over cells 2 and 0, and contains the last two GB of memory.

In one embodiment, the cell controller acquires more information from a cell map entry item than just a cell ID. For instance, there could be several cell

10   map entry items that direct the cell controller to cell 0. The memory contained in cell 0 may have been broken out among various interleaving entries. In one embodiment, a cell map entry item also supplies the cell controller with an indication of where within cell 0's memory the required information is stored. In one form of the invention, a CELL_ADDR, or cell address, is used to direct the

15   cell controller to a specific five hundred and twelve megabyte (MB) region within a cell memory for specific memory access. One embodiment of a cell address equation is provided in the following Equation III:

Equation III

CELL_ADDR = ((ADDR [42:29] & CHUNK_MASK) >>
interleave + CHUNK_START

Address bits twenty-nine through forty-two are used in determining
5   where to start reading data from in a specific cell. The CHUNK_MASK
contained in Equation III is the inverse of the mask previously discussed. The
purpose of the CHUNK_MASK is to direct the cell controller to the right portion
of a block of memory. The masked address portion is right shifted (represented
by the symbol ">>") to eliminate bits that have been used for interleaving
10   (represented in Equation III as "interleave"). If the cell map table had contained
eight columns, three interleaving bits would be used to identify a column. If the
cell map had sixteen columns, four bits would be used for identifying a column.

CHUNK_START is the amount of memory that has been dedicated or
previously used by other interleaving entries. For example, referring back to
15   interleaving entry 1 of Figure 2, interleaving entry 1 came after interleaving
entry 0 had occupied two GB from each of the four cells. Therefore, if the
second two GB portion of cell 0 is being addressed, the CHUNK_START would
be the two GB which were contained in interleaving entry 0. Alternatively, still
referring to Figure 2, interleaving entry 5, which contained two GB in cell 0 and
20   two GB in cell 1, would have a CHUNK_START of four GB in both cell 0 and
cell 1. In summary, the CELL_ADDR equation uses address bits from the
memory accessing device, adds an amount of memory equal to the offset to
arrive at the specific memory location within a given block, right shifts to
eliminate bits that have already been used in interleaving and are no longer
25   necessary to determine the location within the block, and adds cell memory that
has been dedicated to interleaving within other interleaving entries.

Figure 9 is a diagram of one embodiment of a cell map 900. Cell map
900 includes module ID table 912 and chunk start table 914, each including a
plurality of entries 904. Each entry 904 includes a plurality of entry items 910.
30   In an alternative embodiment, the module ID information and chunk start

14

information may be incorporated into a single table. In another alternative embodiment, cell map 900 does not include a chunk start table 914. The number of entry items 910 in each entry 904 is based on the number of ways of interleave. For example, for sixteen-way interleaving, module ID table 912 and

5   chunk start table 914 will each include sixteen entry items 910 per entry 904 (as shown in Figure 9). And for sixty-four-way interleaving, module ID table 912 and chunk start table 914 will each include sixty-four entry items 910 per entry 904. In one embodiment, each entry item 910 in the module ID table 912 includes an eight-bit module ID value for identifying a destination module, and

10   each entry item 910 in the chunk start table 914 includes a chunk start value, which is used to identify a memory location. Based on each received incoming address 902, an appropriate module ID value 916 and chunk start value 918 are identified and output from cell map 900 as described below.

The following Table I includes pseudocode for implementing one

15   embodiment of a cell map based system:

Table I

1.	Incoming_Address[49:0]	
2.	Address Match[49:29]	
3.	Address Mask[49:29]	
4.	Module_id_table_in_entry[Max_ways of interleave in entry * Bits_per_module_id]	
5.	Chunk_start_table_in_entry[Max_ways of interleave in entry * Number of bits required to address all 0.5GB chunks]	
6.	Hit_for_that_entry = ((Incoming_Address[49:29]] & Address Mask[49:29]) == Address Match[49:29])	
7.	Physical_destination_module = Module_id_table_in_entry [Incoming_Address[9:6]]	
8.	Chunk_start = Chunk_start_table_in_entry [Incoming_Address[9:6]]	

For the above Table I, it is assumed that: The address size implementation is fifty bits (i.e., bits zero through forty-nine); each entry 904 is sixteen-way interleaved (therefore requiring four bits for indexing inside an

35   entry 904); the system is performing single cache line interleaving; the cache line size is sixty-four bytes (so address bits six through nine are used for

indexing inside an entry 904); there will be approximately thirty-two to sixty-four entries 904 provided at each processor interface; and the minimum addressable memory is a half of a GB (so the Address Mask uses bits twenty-nine through forty-nine).

5   It will be understood that the above assumptions are based on one example system, and that modifications may be made based upon the particular implementation. For example, if the cache line size were one hundred and twenty-eight bytes rather than sixty-four bytes, bits seven through ten of the incoming address 902 would be used for indexing inside an entry 904. If the

10   cache line size were two hundred and fifty-six bytes, bits eight through eleven of the incoming address 902 would be used for indexing inside an entry 904. Each doubling of the cache line size would shift each of the four indexing bits to the left one bit position. As another example, if the minimum addressable memory were sixty-four megabytes (MB) rather than half a GB, the Address Mask would

15   use bits twenty-four through forty-nine rather than twenty-nine through forty-nine. In addition to different cache line sizes and different minimum addressable memory sizes, other changes may be made, including, but not limited to, different address sizes, multiple cache line interleaving rather than single cache line interleaving, and a different number of ways of interleave.

20   The first item listed in Table I is Incoming_Address[49:0], which is represented in Figure 9 as incoming address 902. In one embodiment, incoming address 902 is a processor address and includes fifty bits, which are numbered zero through forty-nine. Incoming address 902 includes first address portion 902A, second address portion 902B, and third address portion 902C.

25   The second item listed in Table I is Address Match[49:29]. In one embodiment, each entry 904 in a cell map 900 is associated with a particular value for Address Match [49:29].

The third item listed in Table I is Address Mask[49:29]. Address Mask[49:29] is used by a cell controller (e.g., cell controller 175) to extract

30   relevant bits from an incoming address 902. In one embodiment, bits twenty-nine through forty-nine of an incoming address 902 are referred to as first

address portion 902A. Cell controller 175 compares the masked first address portion 902A to Address Match values to identify a matching entry 904.

An entry 904 in module ID table 912 is represented by the fourth item listed in Table I, which is Module_id_table_in_entry [Max_ways of interleave in

5   entry *Bits_per_module_id]. The value inside the brackets for Module_id_table_in_entry represents the bit size of entries 904 in module ID table 912. Assuming that eight bits are used for each module ID (i.e., Bits_per_module_id = 8) and that entries 904 are sixteen-way interleaved (i.e., Max_ways of interleave in entry = 16), the number of bits required for each

10   entry 904 in the module ID table 912 will be one hundred and twenty-eight (i.e., 8 x 16). For sixty-four-way interleave entries 904, the size of module ID table entries 904 will be five hundred and twelve bits (i.e., 64 x 8). Using eight bits per module ID allows unique identification of two hundred and fifty-six modules. Other numbers of bits per module ID may be used depending upon the

15   particular implementation.

An entry 904 in the chunk start table 914 is represented by the fifth item listed in Table I, which is Chunk_start_table_in_entry[Max_ways of interleave in entry * Number of bits required to address all 0.5GB chunks]. The value inside the brackets for Chunk_start_table_in_entry represents the bit size of

20   entries 904 in the chunk start table 914. Assuming that eight bits are used to address all half-gigabyte chunks, and that entries 904 are sixteen-way interleaved (i.e., Max_ways of interleave in entry = 16), the number of bits required for each entry 904 in the chunk start table 914 would be one hundred twenty-eight (i.e., 8 x 16). For sixty-four-way interleave entries 904, the size of chunk start table

25   entries 904 will be five hundred twelve bits (i.e., 64 x 8).

The sixth item listed in Table I is Hit_for_that_entry = ((Incoming_Address[49:29]] & Address Mask[49:29]) == Address Match[49:29]). In one embodiment, each entry 904 in cell map 900 is associated with a particular value for Address Match[49:29], and a hit equation like that

30   listed in the sixth item of Table I is performed for each entry 904 in the cell map 900 until a hit is identified. The variable Hit_for_that_entry will contain either a

logical true value or a logical false value depending on the comparison of (Incoming_Address[49:29]] & Address Mask[49:29]) and Address Match[49:29]. If the value for (Incoming_Address[49:29]] & Address Mask[49:29]) and the value for Address Match[49:29] are not equal,

5   Hit_for_that_entry will be a logical false value, indicating a non-matching entry. If the value for (Incoming_Address[49:29]] & Address Mask[49:29]) and the value for Address Match[49:29] are equal, Hit_for_that_entry will be a logical true value, indicating a matching entry.

For each entry 904 in module ID table 912, there is a corresponding entry

10   904 in chunk start table 914. If a hit is generated for an entry 904, a module ID value 916 from the module ID table entry is output, and a chunk start number 918 from the corresponding chunk start table entry is output. In one embodiment, the chunk start number 918 is a half-gigabyte multiple for identifying a memory location within the memory identified by the module ID

15   value 916. The equations for identifying the appropriate module ID value 916 and chunk start number 918 within an entry 904 are listed in the seventh and eighth items of Table I.

The seventh item listed in Table I is Physical_destination_module = Module_id_table_in_entry [Incoming_Address[9:6]]. As indicated by this item,

20   bits six through nine of an incoming address 902 are used to identify one of the sixteen entry items 910 in a matching entry 904 in the module ID table 912. In one embodiment, bits six through nine of the incoming address 902 are referred to as third address portion 902C. The module ID value from the identified entry item 910 is stored in the variable Physical_destination_module. For a module

25   ID table 912 with sixty-four-way interleave entries 904, the equation for indexing into a module ID table entry 904 would be:
Physical_destination_module = Module_id_table_in_entry [Incoming_Address[11:6]]. In this case, third address portion 902C includes address bits six through eleven, which are used for indexing into a module ID

30   table entry 904.

The eighth item listed in Table I is Chunk_start =
Chunk_start_table_in_entry [Incoming_Address[9:6]]. As indicated by this
item, the third address portion 902C of an incoming address 902 is used to
identify one of the sixteen entry items 910 in a matching entry 904 in the chunk

5   start table 914. The chunk start number from the identified entry item 910 is
stored in the variable chunk_start. For a chunk start table 914 with sixty-four-
way interleave entries 904, the equation for indexing into a chunk start table
entry 904 would be: Chunk_start = Chunk_start_table_in_entry
[Incoming_Address[11:6]]. In this case, third address portion 902C includes

10   address bits six through eleven, which are used for indexing into a chunk start
table entry 904. As described above, in one embodiment, the chunk start value is
used in a cell address equation (e.g., Equation III) to identify a specific memory
location.

The number of entries 904 and the number of ways of interleaving in a cell

15   map 900 are typically selected based on the system size and topology. In a typical
system, a cell map 900 usually includes about thirty-two to sixty-four interleave
entries 904, with each entry being either sixteen-way or sixty-four-way
interleaved. If the flexibility provided by sixty-four-way interleaving was desired,
previous techniques required the use of four times the number of flip-flops needed

20   for sixteen-way interleaving. Such previous techniques were either inefficient in
resource utilization, or limited in capability.

One embodiment of the present invention provides higher-way interleave
entries by combining multiple lower-way interleave entries 904. It will be
assumed in the following description that the baseline interleave entry size is

25   sixteen-way, and that the higher-way interleave entry sizes range from thirty-two-
way to five hundred and twelve way. It will be understood that in alternative
embodiments other interleave entry sizes may be provided. In one embodiment,
higher-way interleaving is achieved by comparing more bits of an incoming
address 902 to find a hit for a given entry 904.

30   The following Table II includes pseudocode for implementing one
embodiment of the present invention:

Table II

	1.	Address Mask [49:29]
	2.	Address Match [49:29]
	3.	Address Mask Lower [15:10]
5	4.	Address Match Lower [15:10]
	5.	Module_id_table_in_entry [Max_ways of interleave in entry * Bits_per_module_id]
	6.	Chunk_start_table_in_entry [Max_ways of interleave in entry * Number of bits required to address all 0.5GB
10		chunks]
	7.	Hit_for_that_entry = TRUE ((Incoming_Address[49:29] & Address_Mask[49:29] == Address_Match[49:29]) && (Incoming_Address[15:10] & Address_Mask_Lower [15:10] == Address_match_lower[15:10]))
15	8.	Physical_destination_module = Module_id_table_in_entry [Incoming_Address[9:6]]
	9.	Chunk_start = Chunk_start_table_in_entry [Incoming_Address[9:6]]

As can be seen from the above Table II, the pseudocode included therein is similar to the pseudocode included in Table I, with the exceptions that Address Mask Lower and Address Match Lower (the third and fourth items in Table II) have been added, and the hit equation (the seventh item in Table II) has been modified. These differences are described below.

The following Table III lists bit values used for Address_Mask_Lower for different interleave sizes according to one embodiment of the invention:

Table III

Address_Mask_lower[15:10] = 000000// 16-way or less interleave
Address_Mask_lower[15:10] = 000001// 32-way interleave
Address_Mask_lower[15:10] = 000011// 64-way interleave
Address_Mask_lower[15:10] = 000111// 128-way interleave
Address_Mask_lower[15:10] = 001111// 256-way interleave
Address_Mask_lower[15:10] = 011111// 512-way interleave

Assume that it was desired to implement a sixty-four-way interleave entry by combining, or grouping together, four sixteen-way entries 904 named ent_a, ent_b, ent_c, and ent_d. Since sixty-four-way interleaving is desired, Address_Mask_lower [15:10] will be 000011 for the four entries, as shown in

20

the above Table III. For all four entries, Address_match [49:29] and Address_Mask[49:29] will be the same. But Address_Match_Lower will be different for each of the four entries. The following Table IV lists values used for Address_Match_lower[15:10] for each of the four entries:

5   Table IV

Address_Match_lower[15:10] for ent_a (1st 16 ways) = 000000
Address_Match_lower[15:10] for ent_b (2nd 16 ways) = 000001
Address_Match_lower[15:10] for ent_c (3rd 16 ways) = 000010
Address_Match_lower[15:10] for ent_d (4th 16 ways) = 000011

10

As shown in the seventh item of Table II, a hit for an entry will be generated if: (1) bits twenty-nine through forty-nine (i.e., first address portion 902A) of the incoming address 902 masked by Address_Mask[49:29] equals Address_Match[49:29]) for the entry 904, and (2) bits ten through fifteen (i.e.,

15 second address portion 902B) of the incoming address 902 masked by Address_Mask_Lower[15:10] equals Address_match_lower[15:10] for the entry 902. The first portion of the hit equation will be true for each of ent_a, ent_b, ent_c, and ent_d. However, the second portion of the hit equation will be true for only one of the four entries. A hit for ent_a will be generated for the zero

20 through fifteen ways of interleave of the full sixty-four-way interleave, a hit for ent_b will be generated for the sixteen through thirty-one ways of interleave of the full sixty-four-way interleave, a hit for ent_c will be generated for the thirty-two through forty-seven ways of interleave of the full sixty-four-way interleave, and a hit for ent_d will be generated for the forty-eight through sixty-three ways

25 of interleave of the full sixty-four-way interleave.

When a hit for an entry 904 is generated, bits six through nine (i.e., third address portion 902C) of the incoming address 902 are used to identify a destination module ID 916 from a module ID table entry and a chunk start value 918 from a chunk start table entry as indicated by the eighth and ninth items in

30 Table II, and as described above.

One form of the present invention allows the size of cell maps to be optimized, while providing expandability. In one embodiment, cell map entries

904 are combinable to form a higher-way interleave entry, which allows logic in the cell controller 175 to be used more efficiently, without sacrificing the ability to interleave more ways. The logic to implement one embodiment of the invention is relatively simple, and only a few more address bits than in previous

5   cell map implementations need to be compared. One form of the present invention provides substantially more flexibility than previous cell map implementations.

Although specific embodiments have been illustrated and described herein for purposes of description of the preferred embodiment, it will be

10   appreciated by those of ordinary skill in the art that a wide variety of alternate and/or equivalent implementations may be substituted for the specific embodiments shown and described without departing from the scope of the present invention. Those with skill in the chemical, mechanical, electro-mechanical, electrical, and computer arts will readily appreciate that the present

15   invention may be implemented in a very wide variety of embodiments. This application is intended to cover any adaptations or variations of the preferred embodiments discussed herein. Therefore, it is manifestly intended that this invention be limited only by the claims and the equivalents thereof.